

To read the article online, visit <http://www.4GuysFromRolla.com/articles/070809-1.aspx>

# Examining ASP.NET's Membership, Roles, and Profile - Part 16

By [Scott Mitchell](#)

## Introduction

Expiring passwords are a common technique used for improving the security of a computer system. In a nutshell, the system specifies a number of days for which a given password is valid, be it 30 days, 90 days, 120 days, or longer. Once this validity window passes, a user must change her password to a new one before being able to log in. Operating systems like Microsoft Windows have long provided such functionality, but password expiry is a feature that is not as prevalent in web applications.

The ASP.NET Membership system and Login Web controls do not offer out of the box support for password expiry. However, all of the pieces are there for us to implement password expiry; it takes just a dash of markup here and a pinch of code there. For example, the Login Web controls include a `ChangePassword` control, which allows a user to reset his password. The Membership system already tracks the last date and time a user changed his password, and with a few lines of code we can customize the authentication logic for the Login control, prohibiting users whose password has expired from logging in.

This article shows how to implement password expiry in an ASP.NET application that uses the `SqlMembershipProvider` Membership provider; the complete code is available for download at the end of this article. Read on to learn more!

## Determining When a User Last Changed Her Password

The [MembershipUser class](#) provides information about a specific user in the Membership system and has properties like `UserName`, `Email`, and `IsApproved`. There's also a [LastPasswordChangedDate property](#), which returns the date and time the user last changed their password. This property is set to the current date and time when the user account is first created or whenever the user changes her password.

The value of `LastPasswordChangedDate` can be access programmatically via a call to the `Membership.GetUser` method. The `Membership.GetUser` method accepts a number of overloads, the two most germane ones for this discussion being: `Membership.GetUser()`, which returns information about the currently logged on user; and `Membership.GetUser(userName)`, which returns information about a specific user. The following code snippet shows how to use the `GetUser` to get information about the currently logged on user and how to calculate the number of days that have elapsed since the user last changed her password.

```
'Get information about the currently logged on user
Dim usrInfo As MembershipUser = Membership.GetUser()

If usrInfo IsNot Nothing Then
    Dim daysSincePwdChange As Integer =
        Convert.ToInt32(DateTime.Now.Subtract(usrInfo.LastPasswordChangedDate).TotalDays)

    ...
End If
```

The code above takes the current date and time (`DateTime.Now`) and subtracts the date and time the

user last changed their password (`usrInfo.LastPasswordChangedDate`). The `DateTime Subtract` method returns a `TimeSpan` object; the `TotalDays` property returns the total number of days (as a `Double`) between the two dates. This `Double` value is then converted into an `Integer` and stored in the `daysSincePwdChange` variable.

### Specifying a Password Expiry

As the above code snippet shows, it's possible to determine how many days ago a user last changed his password. Following the declaration and assignment of the `daysSincePwdChange` variable we could have a conditional statement that looked something like:

```
If daysSincePwdChange > 30 Then
    ... Password expired! ...
End If
```

This code would certainly determine whether the currently logged on user's password had expired, but it introduces a "magic number," 30. Rather than hard coding the expiry let's instead define the expiry in `Web.config`. Having the expiry defined in `Web.config` makes the expiry easier to modify; it also makes the application code more maintainable, as the expiry is defined in a single place rather than being hard coded in (potentially) multiple files.

The `Web.config` available for download at the end of this article includes an `<appSettings>` section with the key `PasswordExpiryInDays` and a value of 30. You can, of course, change this value to whatever expiry best suites your application.

```
<configuration>
  <appSettings>
    <add key="PasswordExpiryInDays" value="30" />
  </appSettings>

  <system.web>
    ...
  </system.web>
</configuration>
```

To facilitate accessing this value from the ASP.NET application I created a helper class named `SecurityUtils` that has a shared, read-only property named `PasswordExpiryInDays`. This property looks inside `Web.config` to see if the `PasswordExpiryInDays` `<appSettings>` value is defined. If so, it returns that value; if not, it returns the value 30.

```
Public Class SecurityUtils
    Public Const DefaultPasswordExpiryInDays As Integer = 30

    Public Shared ReadOnly Property PasswordExpiryInDays() As Integer
        Get
            Dim expiry As String = ConfigurationManager.AppSettings("PasswordExpiryInDays")
            If String.IsNullOrEmpty(expiry) Then
                Return DefaultPasswordExpiryInDays
            Else
                Return Convert.ToInt32(expiry)
            End If
        End Get
    End Property
End Class
```

For more on specifying application settings in `Web.config` see [K. Scott Allen's](#) article, [AppSettings In Web.config](#).

## Preventing Users With Expired Passwords From Signing In

To enforce the password expiry we need to update the login page so that users who have an expired password cannot sign on without first updating their password. [Part 6](#) of this article series showed how to customize the Login control and override its default authentication logic. In a nutshell, this is accomplished by creating an event handler for the Login control's [Authenticate event](#) and determining whether or not the user's credentials are valid.

What we need to do is create an event handler for this event that checks the user's credentials and how long it's been since they last changed their password. If the user's credentials are invalid then we need to indicate that authentication failed. If the user entered valid credentials and has a current password, we indicate that the user has been successfully authenticated. Finally, if the user's credentials are valid but their password is expired then rather than signing the user into the site we instead need to redirect them to a page where they can change their password. This following event handler code implements this logic:

```
Protected Sub myLogin_Authenticate(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.AuthenticateEventArgs) Handles myLogin.Authenticate
    'Are the credentials valid?
    If Membership.ValidateUser(myLogin.UserName, myLogin.Password) Then
        'Has the password expired?
        Dim usrInfo As MembershipUser = Membership.GetUser(myLogin.UserName)

        Dim daysSincePwdChange As Integer =
Convert.ToInt32(DateTime.Now.Subtract(usrInfo.LastPasswordChangedDate).TotalDays)
        If daysSincePwdChange > SecurityUtils.DefaultPasswordExpiryInDays Then
            'Password expired, send user to change password
            Response.Redirect("~/ChangePassword.aspx?UserName=" &
Server.UrlEncode(myLogin.UserName))
        Else
            e.Authenticated = True 'Credentials valid & password is current
        End If
    Else
        e.Authenticated = False 'Invalid!
    End If
End Sub
```

Users who attempt to sign in with an expired password are actually not signed in, but instead redirected to `ChangePassword.aspx?UserName=userName`.

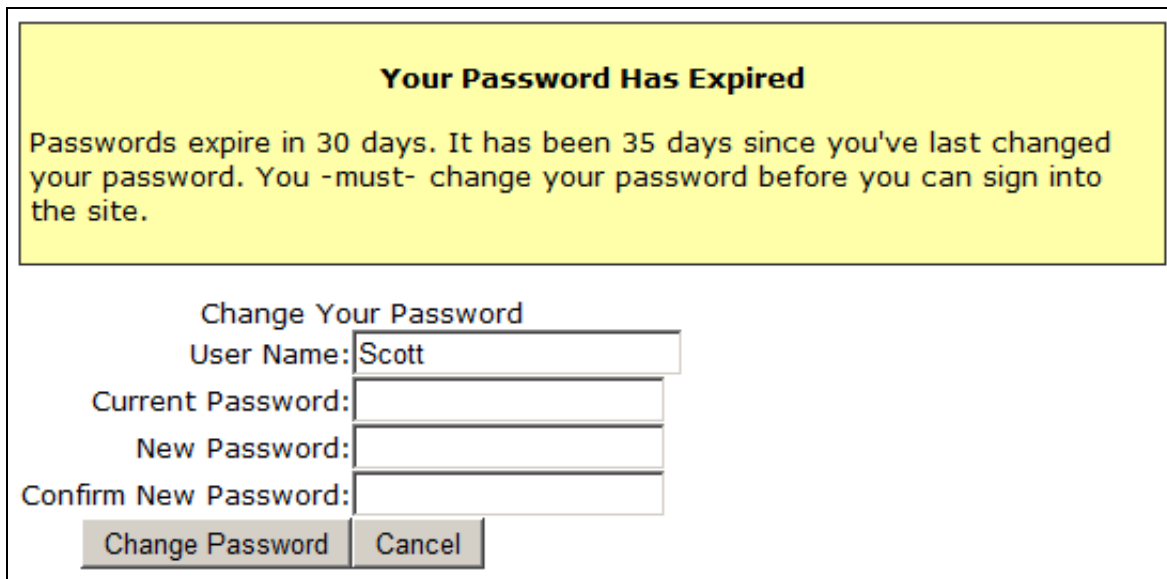
## Creating the Change Password Page

The `ChangePassword.aspx` page uses the `ChangePassword` Web control to provide a user interface that allows the user to change their password. Keep in mind that this page may be visited by anonymous users who are automatically sent here from the login page or by authenticated users who have already signed into the site with an active password, but who want to change their password prior to its expiration.

The `ChangePassword` control has a `DisplayUserName` property that indicates whether the control should include a `TextBox` for the user to type in his username. When an anonymous user visits this page they will need to enter their username along with their old and new password; however, when an authenticated user visits this page, there's no need for them to enter their username because we already know who they are. Consequently, this property is set to `False` if the person visiting the page is authenticated and `False` if they are anonymous. Also, if an anonymous person visits this page and there is a `UserName` value passed through the querystring, then it reasons that the person was sent to this page after attempting to sign in. In such a case a `Panel` is displayed which informs the visitor that they need to change their password because it has expired. Moreover, the `UserName` `TextBox` is pre-

populated with the Username value passed through the querystring.

The following screen shot shows the `ChangePassword.aspx` page as it appears when reached because a user attempted to sign in with an expired password. Note that the page informs Scott that passwords expire in 30 days and that it has been 35 days since his password was last changed. Furthermore, Scott's username is pre-populated in the Username TextBox.



**Your Password Has Expired**

Passwords expire in 30 days. It has been 35 days since you've last changed your password. You -must- change your password before you can sign into the site.

Change Your Password

User Name:

Current Password:

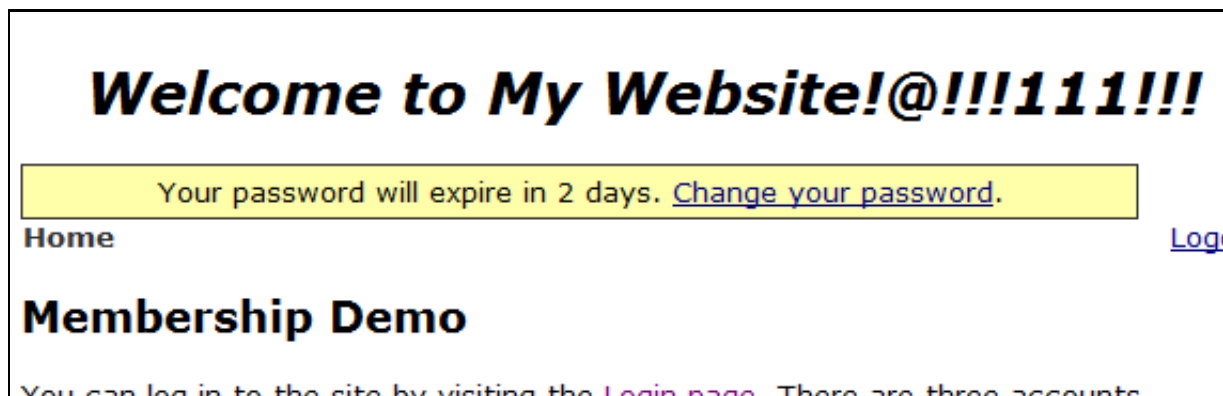
New Password:

Confirm New Password:

At this point Scott can would enter his current password and the new password and click the "Change Password" button. Doing so would update Scott's password, would update the `LastPasswordChangedDate` value for Scott, and would sign Scott into the site. All of this is handled for us automatically by the `ChangePassword` control.

### Displaying a Warning To Users With Passwords About to Expire

The final touch for a system with a password expiry is to notify the user when their password is nearing expiration. To accomplish this I added a Panel to the master page that informs the user how many days exist before their password expires and provides a link to the `ChangePassword.aspx` page. This Panel is only shown if the user's password is set to expire within the next three days.



**Welcome to My Website!@!!!111!!!**

Your password will expire in 2 days. [Change your password.](#)

[Home](#) [Log](#)

**Membership Demo**

You can log in to the site by visiting the [Login page](#). There are three accounts

Because this Panel is in the master page it will appear on every page that uses the master page until the user changes their password. Another option would be to show this message to the user immediately after they signed in (rather than on every page once they've signed in). This could be done by going to the login page and creating an event handler for the Login control's `LoggedIn` event. In this event handler you would determine how many days exist before the user's password expires. If there were three days or less you could send them to a page that explained that their password was set to expire in less than three days, with a link to the `ChangePassword.aspx` page where they could go and update

their password.

## Conclusion

Most user account-based systems have an option that allows the administrator to set a password expiry. While the ASP.NET Membership system does not include an out of the box tool for setting a password expiry, it does provide all of the necessary pieces for implementing such a policy. As this article showed, it is possible to expire passwords in an ASP.NET Membership system with just a bit of code.

Happy Programming!

- By [Scott Mitchell](#)
- 

## Further Reading

- [AppSettings In web.config](#)
- [Website Security Tutorials](#) (VB and C# versions available)

## Attachments

- [Download the code used in this article](#)

Article Information	
Article Title:	ASP.NET.Examining ASP.NET's Membership, Roles, and Profile - Part 16
Article Author:	Scott Mitchell
Published Date:	July 8, 2009
Article URL:	<a href="http://www.4GuysFromRolla.com/articles/070809-1.aspx">http://www.4GuysFromRolla.com/articles/070809-1.aspx</a>

---

Copyright 2014 QuinStreet Inc. All Rights Reserved.  
[Legal Notices](#), [Licensing](#), [Permissions](#), [Privacy Policy](#).  
[Advertise](#) | [Newsletters](#) | [E-mail Offers](#)